Reti di Calcolatori T
Appello del 28/01/2022
Compito 3

Cognome: ……
Nome: ……
Matricola: ……

Tempo a disposizione: 3h

È obbligatorio inserire Cognome, Nome, e numero di Matricola all'inizio di ogni file sorgente, pena la non valutazione del compito, che viene stampato in modo automatico solo in caso siano presenti gli elementi detti sopra.
Si devono consegnare **singolarmente tutti i file sorgente e tutti gli eseguibili prodotti** (*per favore, solo quelli relativi ai file sorgente consegnati!!!*).

La prova intende valutare le capacità progettuali e di programmazione sia in **ambiente Java** che in **ambiente C**, pertanto è consigliato sviluppare **entrambe** le soluzioni richieste al meglio.
In entrambi gli esercizi, sia in Java che in C, si effettuino gli opportuni controlli sugli argomenti della richiesta e si gestiscano le eccezioni verso l'utente, tenendo presente i criteri secondo cui si possa ripristinare il funzionamento del programma oppure si debba forzarne la terminazione.
Leggete con attenzione le specifiche del problema prima di impegnarvi "a testa bassa" nello sviluppo delle singole parti. Naturalmente, ci aspettiamo che i componenti da consegnare siano stati provati e siano funzionanti.

****

Si richiede la progettazione e la realizzazione di **servizi invocabili** da parte di più clienti **sul file system di una macchina server** (direttorio remoto) considerando i file testo residenti sul server.
Ogni comando considera come direttorio di partenza il direttorio corrente, dove sono stati lanciati rispettivamente il client e il server. Se per esempio il client richiede la cancellazione di un file (usando una notazione relativa) nel direttorio corrente, il server cerca di cancellare il file nel direttorio da dove è stato lanciato.
Si considerino solo file di testo e, in particolare, si realizzino i seguenti servizi:

1. **Analisi delle occorrenze di una linea all'interno di un file di testo:** questa operazione richiede all'utente un nome di un file e la linea di interesse (lunghezza massima 256), conta tutte le occorrenze della linea all'interno del file, e restituisce a console l'esito dell'operazione.
2. **Eliminazione di tutte le occorrenze di una specificata parola all'interno di un file di testo:** questa operazione richiede all'utente il nome di un file e la parola, elimina tutte le occorrenze della parola sul file remoto (come separatori ammessi per l'identificazione di una parola si considerino i soli caratteri ' ' e '\n'), e visualizza sul cliente a video il numero di eliminazioni effettuate dal server.
3. **Lista dei file di un direttorio i cui nomi iniziano con un prefisso (inteso come insieme di caratteri)**: questa operazione richiede all'utente un nome di direttorio e un prefisso (max 10 caratteri), quindi visualizza la lista dei file il cui nome inizia col prefisso indicato.
4. **Trasferimento dal server al client di tutti file di un direttorio che contengono un numero di byte maggiore di una soglia**: questa operazione richiede all'utente il nome del direttorio e il valore della soglia e trasferisce i file (get di quel direttorio dal server) che contengono un numero di byte maggiore di quello richiesto, trasferendoli con nome e contenuto sul direttorio del client.

Si richiede inoltre di **non** usare nella soluzione comandi Unix (es. il comando **ls**), ma solo primitive di sistema (es. **opendir**() in C) e funzioni di libreria (es. metodi della **classe File** in Java).

Parte C

Utilizzando RPC sviluppare un'applicazione C/S che consenta di effettuare le **operazioni remote** per:

- contare il **numero di occorrenze di una linea** specificata all'interno di un file di testo;
- ricevere la **lista dei file di un direttorio i cui nomi iniziano con un prefisso**.

Il progetto prevede che il server metta a disposizione due procedure (parte di una interfaccia specificata in XDR e contenuta nel file *RPC_xFile.x*) invocabili in remoto dal client:

- La procedura **conta_occorrenze_linea** accetta come parametro d'ingresso una struttura dati contenente **il nome del file** e **la linea**, conta tutte le **occorrenze della linea nel file**, e restituisce un intero positivo con il numero di occorrenze trovate (>=0) in caso di successo, oppure -1 in caso di insuccesso, ad esempio errori o file inesistente.
- La procedura **lista_file_prefisso** accetta come parametro d'ingresso una struttura dati contenente **il nome del direttorio** e **il prefisso**, e restituisce la lista **dei primi N** (N<=6) nomi **dei file trovati, se ci sono**. In caso di direttorio inesistente, prevedere una segnalazione di errore.

Si progettino inoltre i sorgenti:

- ***RPC_Server*** (contenuta nel file *RPC_Server.c*), che implementa le procedure del server invocabili in remoto;
- ***RPC_Client*** (contenuta nel file *RPC_Client.c*), il processo filtro che realizza l'interazione con l'utente, **propone ciclicamente i servizi** che utilizzano le due procedure remote, e stampa a video i risultati, fino alla fine dello stream di input.

Parte Java

Sviluppare un'applicazione C/S basata su **socket stream** che realizzi le operazioni remote per:

- **eliminazione** di tutte le **occorrenze di una specificata parola** all'interno di un **file di testo**,
- **trasferimento dal server al client** di tutti i file di un direttorio **che contengono un numero di byte maggiore di una soglia**,

utilizzando un'**unica connessione per ogni sessione cliente**: questo vincolo è da **intendersi come prioritario** e **fondamentale**.

Più in dettaglio:

- Il **cliente** è organizzato come un **processo filtro ciclico che consuma l'input fino a fine file** e, per ogni iterazione del ciclo, chiede all'utente quale tipo di operazione vuole effettuare e realizza le interazioni col server utilizzando **una sola connessione per la intera sessione**; alla ricezione del fine file, libera opportunamente le risorse e termina. Per ogni richiesta ricevuta dall'utente, il client prima invia il tipo di servizio al server, poi gestisce gli invii e le ricezioni necessarie alla realizzazione dello specifico servizio richiesto.
  Nel caso della funzionalità di **eliminazione** di **tutte le occorrenze di una specificata parola** all'interno di un **file di testo**, il client invia per ogni richiesta il nome del file e la parola, e riceve l'esito del numero di eliminazioni fatte stampandolo a video.
  Nel caso della funzionalità di **trasferimento dal server al client** di tutti i file di un direttorio **che contengono un numero di byte maggiore di una soglia**, per ogni richiesta, il client chiede all'utente il nome del direttorio e il valore della soglia (un intero che rappresenta il numero di byte), quindi riceve dal server i file di quel direttorio che contengono un numero di byte pari a quello richiesto, e li memorizza con i nomi e contenuto corretto nel proprio file system.
- Il **server** è organizzato come un **unico processo che gestisce in modo parallelo** l'interazione coi clienti, generando un figlio per tutta la sessione di richieste da quel client. Per ogni richiesta, il processo figlio che serve la sessione con una prima lettura discrimina il tipo di funzionalità richiesto, poi gestisce opportunamente l'operazione e si pone in attesa di nuove richieste dallo stesso client; alla lettura della fine sessione, il figlio termina.
  Per ogni richiesta di **eliminazione** di **tutte le occorrenze di specificate parole** all'interno di un **file di testo**, il figlio riceve il nome del file e la parola, quindi effettua l'operazione di eliminazione sul file locale (come separatori ammessi per l'identificazione di una parola si considerino i soli caratteri ' ' e '\n') e invia la risposta al client, rappresentata da un intero positivo che indica il numero di eliminazioni effettuate (>=0) in caso di successo, -1 in caso di problemi.
  Per ogni richiesta di **trasferimento dal server al client** di tutti i file di un direttorio **che contengono un numero di byte maggiore di una soglia**, il figlio legge il nome del direttorio e il valore della soglia (numero di byte), quindi invia nome e contenuto di tutti i file al client.